CHAPTER SUBMITTED FOR INCLUSION IN *COORDINATION THEORY AND COLLABORATION TECHNOLOGY*, EDS: OLSON, SMITH AND MALONE

Seeding, Evolutionary Growth and Reseeding: The Incremental Development of Collaborative Design Environments

Gerhard Fischer[1], **Jonathan Grudin**[2], **Raymond McCall**[3], **Jonathan Ostwald**[1], **David Redmiles**[2], **Brent Reeves**[4], and **Frank Shipman**[5]

[1]Department of Computer Science and Institute of Cognitive Science, University of Colorado, Boulder, Colorado 80309

[2]Information and Computer Science Department, University of California, Irvine CA 92717

[3]College of Environmental Design and Institute of Cognitive Science, University of Colorado, Boulder, Colorado 80309

[4]TwinBear Research, 6138 Gale Drive, Boulder, Colorado 80303

[5]Department of Computer Science, Texas A&M University, College Station, Texas 77849

1. Introduction

For a number of years we created software-based design environments solely to support individual designers. Recently, however, we turned our attention to the problem of supporting long-term collaboration. This takes place when an artifact functions and is repeatedly redesigned over a relatively long period of time— e.g., many years. Such artifacts are increasingly common in a wide range of domains, including the design of buildings, space-based habitats, software, and computer networks— to name a few that we have looked at.

Our initial plan was to modify our previous architecture of design environments to add more support for the evolutionary development of the design and the knowledge about the design. Because we anticipated this would create "messy" information we proposed a process of seeding, evolutionary growth and reseeding of the information in the design environment. In the course of working with network designers, building and evaluating prototype systems, and revising our initial theories of collaborative design practice we developed some new outlooks that we think are quite useful. This chapter discusses these results.

This chapter begins by describing our view of design and long-term indirect collaboration. We then discuss the role that knowledge plays in collaborative design. Next the three phases of seeding, evolutionary growth, and reseeding are each described in detail. A discussion section describes experiences from observing and working with network designers and how these experiences affected our system prototypes. After reviewing related work, we then conclude with the lessons we learned and believe will be of value to others interested in supporting long-term asynchronous design.

2. Long-term Collaborative Design

Teamwork is playing a larger role in design projects [Hackman, Kaplan 74; Johansen 88; DeMarco, Lister 87]. Such projects are increasingly large, complex, and long in duration. The design process takes place over many years, only to be followed by extended periods of maintenance and redesign. Specialists from many different domains must coordinate their efforts despite large separations of distance and time. In such projects, constructive collaboration is crucial for success yet difficult to achieve. This difficulty is due in large part to ignorance by individual designers of how the decisions they make interact with decisions made by other designers. A large part of this, in turn, consists of simply not knowing what has been decided and why.

Meetings and other types of direct communication are the commonly used means for coordination and collaboration in design projects, but in many situations — especially ones involving long-term collaboration — these are not feasible. Design projects that extend over many years can involve a high turnover in personnel. Much of the design work on systems is done as maintenance and redesign, and the people doing this work are often not members of the original design team. But to be able to do this work well, or sometimes at all, requires ‘‘collaboration’’ with the original designers of the system. People who are not in the project group at the same time need to collaborate in long-term design.

2.1. Asynchronous Communication

Much research in supporting collaborative work has gone toward supporting synchronous communication, e.g. COLAB [Stefik et al. 87] and GROVE [Ellis, Gibbs, Rein 91]. Above we argue that long-term collaborative design demands support beyond synchronous communication. Even when it is possible for collaborating designers to have direct communication, there is still much potential in providing tools primarily intended for asynchronous use [Hollan, Stornetta 92].

The primary distinctions between synchronous and asynchronous communication are taken from the Computer Supported Cooperative Work field (CSCW), and are represented by the matrix in Figure 1. The following two examples illustrate the benefits of technologically supported asynchronous communication: voice mail and electronic mail.

Consider the installation of voice mail systems in large corporations. Experience suggests that once people become used to the idea of asynchronous communication by phone, they make better use their phone communication. At first people leave messages like:

> Hi, this is Denny, I guess you’re out, so call me back.

But after a while, people begin to leave messages that contain more than just, ‘‘call me back.’’ For example:

> Hi, this is Denny. I’ve found a problem with the AR 30 report. The due dates are incorrect for customer number 899902. We need to get these right before Friday’s close.

Though it takes time, people learn that it is more efficient to place an item on another person’s ‘‘electronic stack’’ or ‘‘inbox’’ than to interrupt what they were doing with a phone call. And though people usually prefer to speak to someone in person, they nevertheless learn how to make good use of the technology. Leaving a detailed message makes more sense for many of the messages and requests. It can be better to give the person time to research the question and call back, rather than surprise him with a problem and expect instant diagnosis.

Asynchronous communication begins to be used where synchronous previously dominated. Though phone mail is at first thought of as an inconvenient backup to the more preferable voice-to-voice, it becomes a useful service in its own right, and more than just a back-up.

Place ↓ / Time →	Same	Different
Same	meeting rooms	work shifts
Different	tele-conferencing	email

Figure 1: Asynchronous Communication

A matrix of CSCW perspectives developed by Johansen [Johansen 88]. Our work focuses on technological support for Asynchronous (i.e., different time, different place) communication. Email is the prototypical example of asynchronous communication.

In arguing against the assumption that the goal of computational media should be to emulate face-to-face communication, Hollan and Stornetta [1992] cite email as the ‘‘paramount success of computationally-mediated informal communication.’’ The interesting aspect related to the point argued here is the statement:

> It meets our critical litmus test of being used by groups even when in close physical proximity. In fact, in our own experience, it is not uncommon to send email to someone in the next adjacent office, or even someone sharing an office.

Like voice mail, in certain situations, email has changed from a tolerable substitute to a preferred medium.

Electronic mail plays an important role as a success model of computer supported communication. Though much CSCW research focuses on providing media which emulate face-to-face meetings, the success of email is a reminder that there is more to good communication support than emulating face-to-face communication. A benefit from asynchronous communication is that it is *archived somewhere*. Whether this is digital voice recording, or electronic mail, it is available for later retrieval. Clearly this does not solve the problem of information retrieval later, but it is a first step.

The two examples, voice mail and electronic mail, illustrate that asynchronous communication comes to be preferred to synchronous communication for certain types of information. But in this chapter we make the stronger case that not only is asynchronous communication an improvement in some cases, it is absolutely necessary for long-term design. To model it only after email or vmail is to underutilize an important resource for design teams. Careful analysis led us to introduce a key distinction in asynchronous communication, that of predictability.

2.2. Long-term, Indirect Communication

In long-term collaborative design tasks, communication between designers is not only asynchronous with respect to time and place, but it is also *indirect* in the sense that the senders and receivers of information are not known a priori. Figure 2 introduces predictability into the well known 2x2 matrix of CSCW (compare Figure 1). Also note that predictability pertains to the participants as well. Long term projects

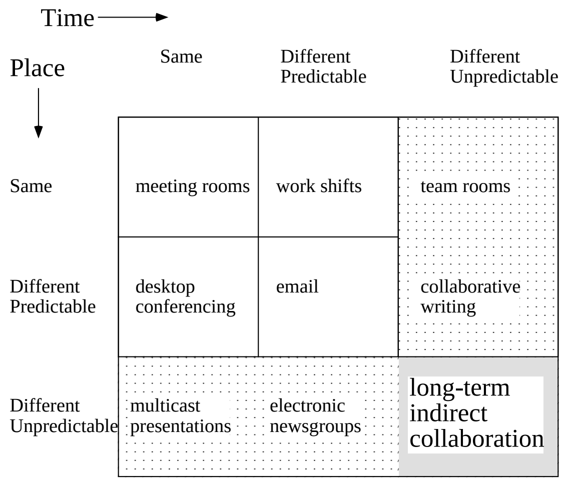

Figure 2: A Classification of Different CSCW Perspectives

This classification scheme extends the matrix shown in Figure 1. Our focus is on the unpredictable communication that occurs throughout the design life-cycle of complex systems. Not only are the time and place unpredictable: the participants themselves are not always known over the long life-cycles of complex systems [Grudin 94a].

are unpredictable with regard to the team members and users who need to communicate. As will be argued later in more detail, this attribute caused us to pursue what we describe as *embedded communication*, where the communication is in a sense ‘‘embedded’’ in the design artifact rather than being stored separately.

Long-term, indirect communication is of particular importance in situations where:

- direct communication is impossible, impractical or undesirable
- communication is shared around artifacts
- designed artifacts continue to evolve over long periods of time (e.g, over months or years).
- designers need to be informed within the context of their work

Support for indirect coordination and collaboration must go beyond what electronic mail and most proposed CSCW software could provide. This support should allow team members to work separately — across substantial distances in space and time — but alert them to the existence of potential interactions between their work and the work of others. Where such interactions exist, support should be provided for collaboration and conflict resolution. Designers must be able to *interact with design artifacts created by previous designers*. Technology enabling this could effectively create virtual cooperation between all designers who ever worked on the project.

These challenges motivated not only a new conceptual model of design environments, but also a new model of the design process.

3. The Evolution of Knowledge in Design

Our conceptual framework grew from systems which try to augment an expert solving a problem, to cooperating experts collaborating over many years and different places.

In the process of our initial research, we formulated a design process model which we now believe is an important aspect of designing systems for collaboration. The process model is motivated by how large software systems, such as GNU Emacs, Symbolics' Genera, Unix, and the X Window System, have evolved over time. In such systems, users develop new techniques and extend the functionality of the system to solve problems that were not anticipated by the system's authors. New releases of the system often incorporate ideas and code produced by users.

In the same way that these software systems are extensible by programmers who use them, design environments need to be extended by domain designers (our term for the users of design environments) who are neither interested nor trained in the (low-level) details of computational environments [Nardi 93].

We illustrate our conceptual framework in the domain of computer network design, which involves complex artifacts that are continuously modified and redesigned. The domain itself is also constantly changing as new technologies are developed.

Knowledge acquisition is a crucial issue in the creation of effective information systems of all types (including expert systems, hypermedia systems, and design environments). There have been two extreme approaches: one is to input information in advance of use, typified by expert systems [Buchanan, Shortliffe 84], and the other is to start with an empty system and allow its information base to grow and become structured as a consequence of use, characterized by initial proposals for argumentative hypertext [McCall, Schaab, Schuler 83; Conklin, Begeman 88]. Neither approach is adequate for the information needs of designers.

The ''put-all-the-knowledge-in-at-the-beginning'' approach fails for numerous reasons. It is inadequate for domains in which the domain knowledge undergoes rapid changes (the computer network domain being a prime example). Traditional knowledge acquisition approaches ,which require domain designers to articulate their knowledge outside the context of problem solving or during an initial knowledge acquisition phase, fail to capture *tacit* knowledge [Polanyi 66], because designers know more than they can tell environment developers. Tacit knowledge is a part of human expertise that surfaces only in the context of solving specific problems.

The ''just-provide-an-empty-framework'' approach requires too much work of designers in the context of a specific project. The difficulties of capturing design knowledge from design projects are well known [Fischer et al. 91]. Documenting interferes with the thinking process itself, disrupting design and requiring substantial time and effort that designers would rather invest in design. Designers typically find it difficult to structure their thoughts in a given format, regardless of the format used [McCall 91]. In addition, domain designers often lack the knowledge and the interest to formalize knowledge so it can be computationally interpreted [Shipman 93].

Our model is between the two extremes of ''put-all-the-knowledge-in-at-the-beginning'' and ''just-provide-an-empty-framework.'' Designers are more interested in their design task at hand than in maintaining the knowledge base. At the same time, important knowledge is produced during daily design activities that should be captured. Rather than expect designers to spend extra time and effort to maintain the knowledge base as they design, we provide tools to help designers record information quickly and without regard for how the information should be integrated with the seed. In our model, knowledge base maintenance is periodically performed by environment developers and domain designers in a collaborative activity.

Our domain-independent design environment architecture plays an important role in the continual development of design environments. It provides a structure for domain knowledge and mechanisms for delivering knowledge as it is needed to support the design task at hand. We have developed our domain-independent architecture through numerous attempts to create domain-oriented design environments [Fischer 92]. The architecture consists of the following five components: (1) a construction component, (2) an argumentation component, (3) a catalog of interesting design examples, (4) a specification component, and (5) a simulation component. The individual components are linked by knowledge-based mechanisms: a construction analyzer (built as a critiquing system [Fischer et al. 93]), an argumentation illustrator, and a catalog explorer [Nakakoji 93]. Design environments contain information encoded using a variety of representational formalities. Construction kits and critics are considered formal representations of design knowledge because they are interpreted by the computer. Argumentation is a semi-formal representation in which informal textual and graphic records are linked by formal associations.

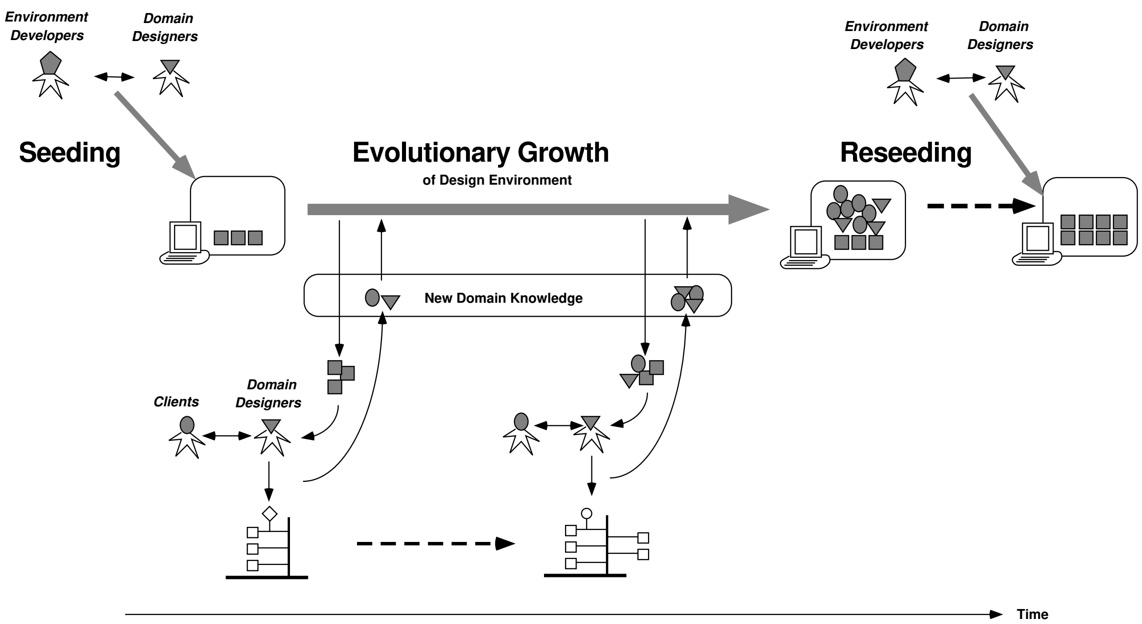

Figure 3: Seeding, Evolutionary Growth, and Reseeding: A Process Model for Domain-Oriented Design Environments

During seeding, environment developers and domain designers collaborate to create a design environment seed. During evolutionary growth, domain designers create artifacts that add new domain knowledge to the seed. In the reseeding phase, environment developers again collaborate with domain designers to organize, formalize, and generalize new knowledge.

Our process model for continual development of design environments from an initial seed through iterations of growth and reseeding is illustrated in Figure 3:

- The *seeding* process, in which domain designers and environment developers work together to instantiate a domain-oriented design environment seeded with domain knowledge.
- The *evolutionary growth* process, in which domain designers add information to the seed as they use it to create design artifacts.
- The *reseeding* process, in which environment developers help domain designers to reorganize and reformulate information so it can be reused to support future design tasks.

To illustrate the evolution of design environments, we discuss seeding, evolution through use, and reseeding in detail in the following three sections. NETWORK [Shipman 93; Fischer et al. 92], a design environment supporting computer network design, is used for illustration.

4. The Seeding Process

A seed is built by customizing the domain-independent design environment architecture to a particular domain through a process of knowledge construction. Although the goal is to construct as much knowledge as possible during seed-building, for complex and changing domains complete coverage is not possible. Therefore, the seed is explicitly designed to capture design knowledge during use [Girgensohn 92].

Domain designers must participate in the seeding process because they have the expertise to determine when a seed can support their work practice. Rather than expecting designers to articulate precise and

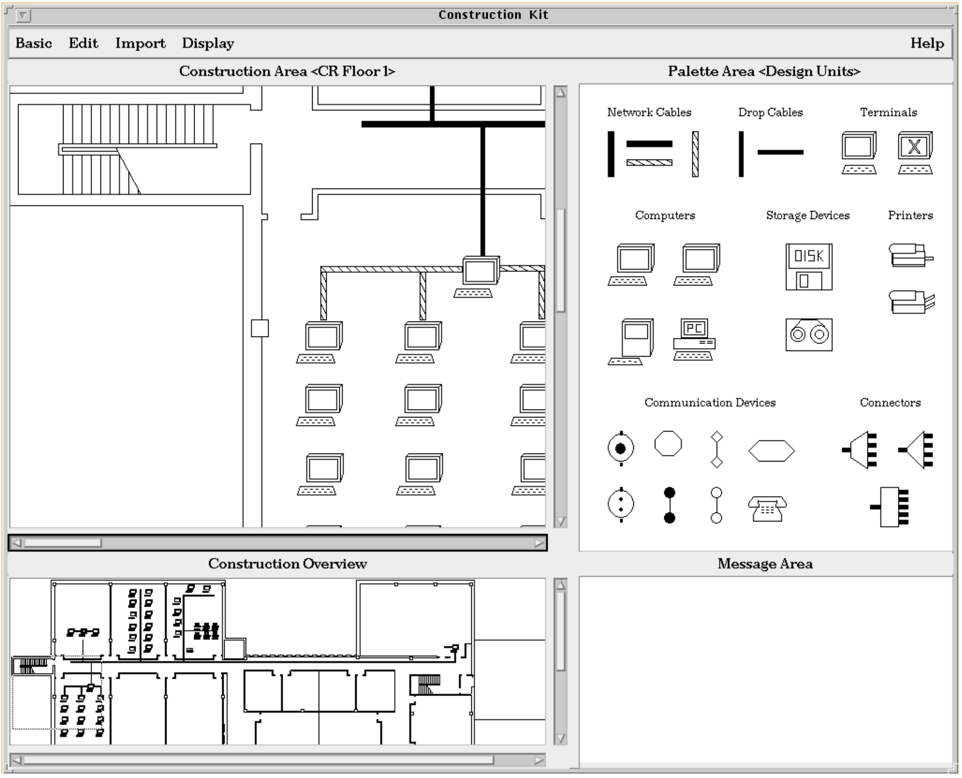

Figure 4: An Environment Supporting Computer Network Design

A screen image of the NETWORK seed. Shown are a palette of network objects (upper right) and the construction area where logical networks are configured (upper left).

complete system requirements prior to seed building, we view seed building as knowledge *construction* (in which knowledge structures and access methods are collaboratively designed and built) rather than as knowledge *acquisition* (in which knowledge is transferred from an expert to a knowledge engineer and finally expressed in formal rules and procedures). New seed requirements are elicited by constructing and evaluating domain-oriented knowledge structures.

The seeding process for the NETWORK design environment (see Figure 4) was driven by observations of network design sessions, prototypes of proposed system functionality, and discussions centered on the prototypes. In design sessions, a logical map of the network being designed served to ground design meetings, discussions, what-if scenarios, and disagreements. The logical map was chosen as the central representation of the artifact in network design, and a prototype construction kit was implemented based on the logical map [Fischer et al. 92]. Evaluation of the NETWORK seed indicated that designers need support for communication in the form of critiques, reminders, and general comments [Reeves, Shipman 92a]. Pointer, annotation, and sketching tools were integrated into the construction kit so talking about the artifact takes place within the artifact representation space.

An important lesson we learned during the seeding of NETWORK was to base our design discussions and prototyping efforts on existing artifacts. Discussing the existing computer science network at CU Boulder was an effective way to elicit domain knowledge because it provided a concrete context that triggered domain designers' knowledge (often in the form of ''war stories''). We found high-level discussions of general domain concepts to be much less effective than discussions focused on existing domain artifacts.

Information to seed NETWORK was acquired from existing databases containing information about net-

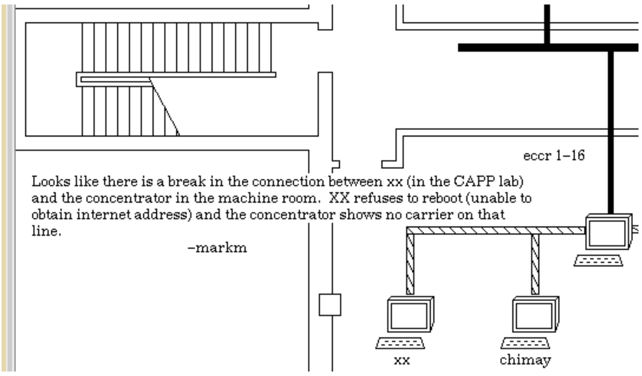

Figure 5: Logical Map with Embedded Discussion

The logical map serves to abstract away low-level details while allowing discussions about the artifact to be embedded in the design.

work devices, users, and the architectural layout of our building. The NETWORK seed contains formal representations of approximately 300 network devices and 60 users. Autocad^{TM} databases created by facilities maintenance personnel provide architectural details of about 100 rooms. This information is represented in NETWORK's construction kit and in the underlying knowledge representation formalisms. The informal part of the NETWORK seed includes notes from the systems administration class, knowledge about the various research groups, and electronic mail of the network designers.

5. Evolutionary Growth Through Use

During the use phase, each design task has the potential to add to the knowledge contained in the system. New construction kit parts and rules are required to support design in rapidly changing domains [Fischer, Girgensohn 90]. Issue-based information in the seed can also be augmented by each design task as alternative approaches to problems are discovered and recorded. The information accumulated in the information space during this phase is mostly informal because designers either cannot formalize new knowledge or they do not want to be distracted from their design task.

Our approach to this challenge is to view the design environment seed as a medium for communication as well as design. Our critique of current design systems is that they function as ‘‘keepers of the artifact,’’ in which one deposits representations of the artifact being designed. But our experience has shown that designers integrate designing and discussing in such a way as to make separate interpretation difficult [Reeves 93]. Talking *about* an artifact requires talking *with* the artifact. Therefore later interpretation of the discussion requires that the discussion be embedded in the context in which it was originally elicited. The most important implication of this view is that design artifacts must not be artificially separated from the communication about them.

This integration was seen in two ways. First, in design sessions videotaped for analysis, *deictic references* (referring to items by the use of ‘‘these,’’ ‘‘those,’’ ‘‘here,’’ etc.) were frequent. A long-term study of network designers showed that users took advantage of embedded annotations and made frequent use of deictic references [Reeves 93]. Second, discussion about the artifact guided the incremental design process. Designers took every opportunity to illustrate critiques of each other's work. Only rarely was a detailed comment made and not accompanied by changing the artifact.

The logical map mentioned above served not only to represent the real network, but also as a medium through which changes were considered and argued (Figure 5). It focused as well as facilitated discussion. Frequently, in arguing over design artifacts, specific issues led to discussions of larger issues. Collaborating designers preferred to ground discussions in design representations. The logical maps served to (1) point out inconsistencies between an appealing idea and its difficulty of implementation, (2) remind participants of important constraints, and (3) describe network states before and after changes.

6. Reseeding

Acquiring design knowledge is of little benefit unless it can be delivered to designers when it is relevant. Periodically, the growing information space must be structured, generalized, and formalized in a reseeding process, which increases the computational support the system is able to provide to designers [Shipman, McCall 94].

The task of reseeding involves environment developers working with domain designers. After a period of use, the information space can be a jumble of annotations, partial designs, and discussions mixed in with the original seed and any modifications performed by the domain designers. To make this information useful, the environment developers work with the domain designers in a process of organizing, generalizing, and formalizing the new information and updating the initial seed.

The Need for Reorganization

The organizational aspect of reseeding is necessary because the information space, through modification by designers using the system, eventually becomes inconsistent. For example, as new technology becomes available, informal notes might contradict formal knowledge about network devices.

When an information space is disorganized, it becomes difficult for designers to locate information that they need. This also makes it more difficult to find the ‘‘right’’ place to add new information, thereby compounding the problem. Disorganization can occur when information about the same topic has become located in separate parts of the information space, or when information has been put in a location where designers cannot find it.

The Need for Generalization

Reuse of information between projects requires the generalization of task-specific information entered during use. The goal is to create more generally applicable information by integrating information about specific situations. This is related to the need for reorganization when variations of the same ideas have been added in project-oriented parts of the information space.

An example of generalization in the network domain is that while documenting changes to a design, information concerning the conversion of sections of a network to a new networking standard will likely appear with each conversion of a subnet. In order to bring this information together into a coherent whole, the subnet-specific details need to be abstracted so that the information is of use in new situations.

The Need for Updating Information

As described earlier, systems supporting rapidly changing domains are forced to evolve or become obsolete. While some of this evolution can occur as evolutionary growth, still there is the need for concerted efforts by environment designers to update domain information.

As an example of the difficulty and potential for support in updating the information consider the problem of updating the initial formal structures created during our seeding of Network. There is frequent change in the number and type of devices, people, and places that the environment needs to represent.

One potential for supporting this process comes from the existence of on-line sources for some of this information. The sources of on-line information individually contain only part of the information needed. Also, the sources use a variety of representations and identifiers for the information. As a result, the process of updating turns out to be heuristic. Still, the automation of changes aids the knowledge engineers in their updating of the information space, requiring them to determine and handle exceptional cases and to disambiguate between multiple potential cross-references.

This experience with Network has shown the need to devise semi-automatic methods for updating information from other on-line sources during reseeding. While much information in computer network design must be on-line, such as user and device profiles, other domains are also likely to have potential on-line sources of information [data-mining]. Support based on these sources can aid both domain designers and environment designers in keeping the design environment up to date.

The Need for Formalizing Information

A final task that is part of reseeding is the formalization of domain information entered during evolutionary growth but not in a format useful for providing knowledge-based support. Experience has shown that in many cases users cannot or will not formalize information on their own [ShipmanMarshall1994].

An example of this is that designers may make comments in notes to themselves or one another about the characteristics of a network device and yet not enter that information into the environment's description of the device. Designers are not being reticent, but cautious in taking time performing tasks which are not necessary to completing their primary task, the design and instantiation of the network.

During reseeding, both because the primary task is the improvement of the information in the design environment and the involvement of the environment designers, the formalization of information entered informally during evolutionary growth becomes possible.

7. Discussion

Experience with the incremental development of the collaborative design environment Network has shown the utility (and perhaps necessity) of using a seeding, evolutionary growth, reseeding process. At the same time it has pointed out the need for further types of support to be provided by design environments for such a process to be successful.

First, the design environment needs to become more than just the storage mechanism for a design, and must start becoming a medium for communication between designers. This leads to the need for embedding communication with the design artifact. Once an artifact and discussion of the artifact develop over some time it become necessary to provide prior context to enable the comprehension of comments and design decisions.

Information must be allowed to enter the environment in an informal representation but formal representations are needed to provide knowledge-based support. Thus, computer support for formalization of knowledge is important for successful design environment development. Because of the size of the information space and the potential for missing relevant information entered by other designers, mechanisms are required to help the location and communication of information. This has led to an initial investigation of computational agents which convey design decisions and opinions to other designers.

7.1. Embedded Communication

Analysis of design sessions in several domains [kitchen, architecture, network] showed that design artifacts ground discussions so strongly that the language itself is difficult to understand without access to video. Deictic references (this, over there, here, that) are frequent, yet computer-based design systems such as CAD, do not support this referencing. This led us to embedded communication: including the discussion about a design artifact in the artifact itself. Though intended to address deixis, this also addressed a shortcoming we see in design rationale systems, namely that the artifact and its rationale are stored separately. This quickly leads to inconsistencies between what was done, and what was supposed to have been done, or what user think should have gotten done.

Competent practitioners usually know more than they can say [Polanyi 66], and conversation leaves many things tacit. One could attempt to overcome these tacit aspects by forcing designers to make more knowledge explicit. Against this approach, which might be labeled the ‘‘tyranny of the explicit’’ [Hill 89], these aspects are seen as motivation for providing computational media in which the designer's natural level of tacit knowledge is respected. The design process suggests the need for a medium in which the design artifact emerges, and which allows the designer to undergo frequent ‘‘shifts in stance’’ [Schoen 83 p. 101].

Observations of collaborating designers using NETWORK show that artifacts serve as medium for communication. Furthermore, discussions about the artifact guide the incremental design process. When communicating asynchronously via textual annotations, network designers integrated the notes and the artifact in ways that made separate interpretation difficult.

7.2. Embedded History

In the same way that evidence of physical history guides cognitive tasks, computational media should provide cues of use which guide design tasks [Hill et al. 92]. For example, as auto parts manuals become worn, they provide visual and tactile cues to guide further use. In the same way that physical wear and tear can be a resource, computational media should embed the history of an artifact in the artifact so that it can serve as guide to further use. Computational media represent the potential to provide history access mechanisms which go beyond what is possible with physical artifacts, such as providing access by various perspectives such as date, user, design change, and relation to other design units.

The approach here was to recognize the fluid nature of the design process and create a computer-based environment in which the artifact is less of an end-product and more of a process. If capturing the design process can be done in a way that does not interfere, then others will be better able to learn from observing the design sequence later. Understanding a complex design is best done by studying the process as well as the product [Kuffner, Ullman 91].

Users draw heavily from past experience in solving current problems [Lee 92]. Computational tools should therefore support this human tendency to reuse previous experience. However, a complicating factor is the tendency for people to ‘‘misremember’’ an event according to plausible inference rather than exact recall [Reder 82]. There is much potential for computer systems to serve as external memory aids in restoring the context surrounding past design decisions [Suchman 87; Anderson 85; Reder 82]. The context becomes all the more important as collaboration increases. In the context of collaborative design, it is not enough to provide *user* history, there should be *artifact* history.

Wolf and Rhyne [1992] argue that the process by which information is created and used can be important for understanding of the end product of a work group. In a study done to gain insight into how to facilitate information retrieval in computer-mediated design sessions, they analyzed how group participants used videotape to access meeting information. They found that people searched for information using four main access methods:

- by participant: they remembered person X doing some action
- by communication medium: people recalled what medium was used (eg. whiteboard, overhead transparency)
- by time: people used relative time ("midway through the meeting"), duration ("25 minutes into the discussion"), and clock time ("we only got through item 1.2 by 5 o’clock")
- by relation to other events: people used events as markers before/after other events.

These findings of how people use videotape for information retrieval serve as challenges for computational history mechanisms.

Hutchins’ [1990] study of team navigation of large ships also motivates history for collaborative artifacts:

> The work a chart does is performed on its surface — all at the device interface, as it were — but watching someone work with a chart is much more revealing of what is done to perform the task than watching someone work with a calculator or a computer [Hutchins 90p. 217]

Asynchronously communicating designers do not have the possibility of ‘‘watching someone work with a chart.’’ However, by keeping the artifact history, the interaction is available for *watching* at a later time. To relate it to Hutchin’s study, imagine a chart which could replay the interaction that took place and show the instruments as they were used.

Design history also provides an approach to design rationale. Though design rationale appears to have great promise [Kunz, Rittel 70], there have been few recorded successes [Yakemovic, Conklin 90]. The designers must perceive a benefit for the extra cost of documenting their reasoning [Reeves, Shipman 92b]. History is therefore a potential candidate for an interaction tool, because there is no extra cognitive cost associated with having history support. Yet it provides the benefit of restoring the context of previous work, others' as well as one's own.

The benefit of history related to design rationale is that in domains such as network design, which involve two-dimensional sketches and graphical representations, designers can often deduce rationale by seeing the process of how something came to be [Kuffner, Ullman 91; ChenDietterichUllman 91]. A logical map of the current network hides many tradeoffs and compromises that were made in the past, yet which still affect current decisions. Having the history of the evolution clarifies some of the tacit knowledge that is represented in the static logical map.

One side benefit of some groupware aids is that they also help the individual. For example, one designer said, "What did I do last?" Though the history was primarily viewed as a tool to help one understand other designers' work, it is also useful for reminding oneself of one's own work, called ‘‘reflexive CSCW’’ [Thimbleby, Anderson, Witten 90]. Usually adding multiuser features complicates the system for single users, but history is an example of both kinds of use.

Context is important in Reminding

Research in human memory has shown that people are prone to recall by inferring "what is plausible given what they can remember" [Anderson 85]. Memory performance improves the more closely the current context matches the past physical, emotional and internal context. Much of recall involves plausible inference rather than exact recall [Reder 82].

History tools are needed to support collaborative design. The motivation for this argument lies in the work done in situated cognition relating to context [Lave 88; Carraher, Carraher, Schliemann 85]. Design environments can capture only a portion of the whole context, namely the dates when a given user made certain changes. Yet this small portion can be important in collaborative design.

Each designer on a project team understands only a portion of the overall design artifact. As large projects evolve over time and turnover and attrition take their toll, it will become increasingly important for computer based design environments to help capture the evolution of an artifact and not just its current state. The history serves to remind designers of how the artifact came to be and what the context was when certain decisions were made.

7.3. Computer Support for Formalization of Knowledge

Study of collaborative design projects showed that designers rapidly vary from informal to formal representations. But current CAD systems, for example, only allow the formal representation to be stored. Yet research is showing the importance of "informal sketches" [Gross1993]. These sketches should be part of the system, because they exert enormous influence throughout the development and implementation process. Rather than making arbitrary distinction between formal and informal data, we support a smooth migration from informal representations such as textual notes and sketches, to more formal representation such as OO class diagrams, or standardized graphical items such as palettes.

To address the difficulties involved in formalizing information by both environment designers and domain designers during all phases of design environment development we have developed tools that support the process of formalization [Shipman 93].

One class of tools suggests possible formalizations that could be added to the information space based upon the variety of information that the system already has available from both the seed and information added during use. The formally represented information, along with the placement, textual content, and textual attribute values, can be used by these tools.

For example, one tool in NETWORK looks for vocabulary in textual values of attributes that might relate to

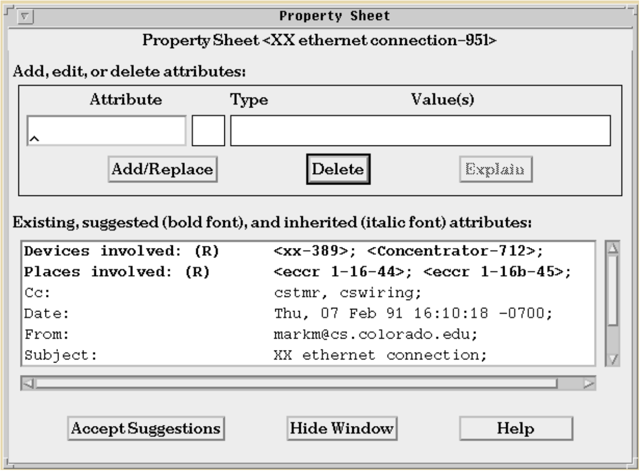

Figure 6: Property Sheet with Suggested Attributes

Suggested attributes for an imported electronic mail message appear in bold in the property sheet. The text of the mail message is shown in Figure 5. When an attribute (e.g., "Devices Involved") is selected by the user, it appears in the top portion of the property sheet, where it may be edited.

other objects and suggests replacement (or augmentation) by a relationship between the two objects. An example of this would be a workstation (c3d2) in the design that has an attribute ‘‘disk server’’ with the value ‘‘c3d1’’ as a text string. This tool would suggest the recasting of this attribute to be a relation, instead of a string, pointing to the object in the design that represents the device c3d1.

Tools can also make use of possible references found in the textual display of objects to suggest new attributes and relations. As an example we will discuss the text annotation in Figure 5, which was taken verbatim from an electronic mail message between network designers.

Recognizing some of the references to concepts already formally represented in the system provides domain- and design-specific cues as to the content of the object. Based on the occurrence of these text references, the system suggests new attributes for the mail message, as shown in Figure 6. In this example, the system has recognized references to devices, and places already known to the system. Further, these new attributes can be used later to locate related information.

7.4. Agents to support communication

When many designers participate in a project, it is difficult for each to know the whole project. How can each be informed of the changes which affect him, and how can a designer find old knowledge that affects a future modification? The issue of how to make information in group memory available to designers presents formidable challenges.

Our prior work on critics [Fischer et al. 93] and information delivery [Fischer, Nakakoji 91] for individual work has provided the potential solution of creating mechanisms which automatically volunteer information. But having the design environment volunteer information leads to issues such as how to determine what information to volunteer, how to volunteer it, how to let designers interact with the environment to determine what information is volunteered to them and to other designers.

In the context of NETWORK, we have begun investigating the use of agents to support communication. Compared to other work on agents providing information we have focussed on how designers, rather than text analysis algorithms combined with complex interest profiles, can directly determine what and how information is volunteered. To do this we have focussed on how designers can create agents.

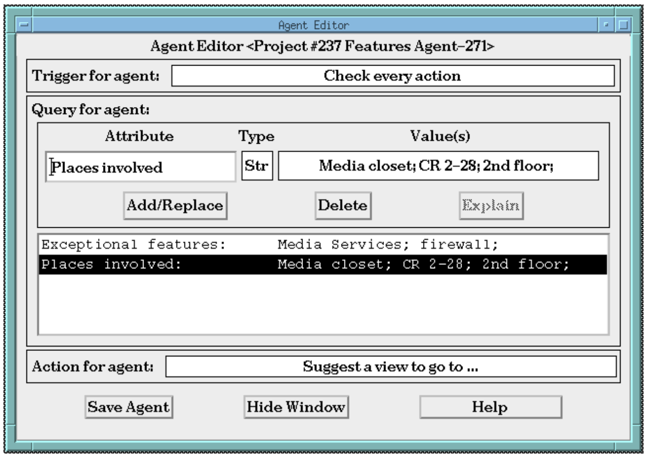

Figure 7: Agent Editor showing Current Status of an Agent

Designers can define and edit agents in NETWORK's agent editor. By selecting a trigger, a query, and an action, designers can decide the information to be displayed, the situation in which to display it, and the manner in which to display it.

As with all the rest of the information in the collaborative design environment, agents are created and evolve through all phases of environment development. An initial set of agents are created during seeding. During the evolutionary growth phase, agents can be created and modified in the agent editor, shown in Figure 7. Finally, like other types of information, agents can be added, edited or removed during reseeding.

8. Relation to other work

We have developed domain-oriented design environments [Fischer 92] to support design in a variety of domains, including user interfaces [Lemke, Fischer 90], buildings [McCall, Ostwald, Shipman 91], computer networks [Fischer et al. 92], and voice dialog systems [Repenning, Sumner 92].

Our previous work on design environments, such as FRAMER [Lemke, Fischer 90], JANUS [Fischer, McCall, Morch 89a], and PHIDIAS [McCall et al. 90] has emphasized systems for individual designers by providing support of human problem-domain communication, communicating via abstractions and concepts specific to a domain [Fischer, Lemke 88], and construction kits. Human problem-domain communication and construction kits are necessary but not sufficient for good design.

Upon evaluating prototypical construction kits [Fischer, Lemke 88], we found that they do not by themselves assist the user in constructing interesting and useful artifacts in the application domain. To do this they need knowledge to distinguish ‘‘good’’ designs from ‘‘bad’’ designs. Design environments combine construction kits with *critics* [Fischer et al. 91]. Critics use knowledge of design principles for the detection of suboptimal solutions constructed by the designer.

One of the challenges for critiquing systems is avoiding work disruption. Our systems accomplish this by making the critics sensitive to the specific design situation [Fischer, Nakakoji 91], incorporating a model of the user [Fischer et al. 91], and giving users control over when and which critics should fire [Fischer, Girgensohn 90]. Part of the JANUS project [Fischer, McCall, Morch 89b] focussed on building critics which are useful in spite of not being highly intelligent (i.e., more intelligent than a designer). Simple critics can be informative because they are based on domain knowledge that designers might not have (e.g., network designers are not necessarily familiar with relevant knowledge about fire codes for buildings).

Problems often arise in the use of collaboration technology when some people ar required to do additional work to support a system that primarily benefits other users [Grudin 94b]. This is particularly important for systems that would incorporate design rationale information, a very difficult task [Fischer et al. 91]. The system and process we have described addresses this in several ways. By dividing the labor between environment designers and domain designers, between the seeding/reseeding activity and the evolutionary growth, we allow different players to focus on their work on tasks for which they will be rewarded. Also, allowing desiginers to comment informally during the desig process, with formalization of the knowledge postponed until later, minimzes the cost to the designers while permitting others to benefit from the eventual inclusion of the knowledge in the system.

This project gave us an opportunity to build upon our previous design environment work and to broaden these environments to investigate complex issues in long-term collaborative design.

The usefulness of our ‘‘seeding — evolutionary growth — reseeding’’ model has been demonstrated for a variety of systems such as operating systems [Walker et al 87] and domain-oriented design environments [Fischer et al. 92].

The model successfully addresses a number of important issues. On the one hand, it is an intentional effort to recognize the importance of specialization at the expense of expressive generality [CSTB 90]. By providing a significant seed of knowledge for domain-oriented design environments, specific design projects do not have to recreate domain-oriented abstractions but only have to extend the seed where it is incomplete or inaccurate for their task. New designs can be described using concepts, rules, and examples contained in the seed. The model avoids the pitfalls of expert systems approaches, which are often built on the assumption that *all* relevant knowledge for a domain can be encoded into a system.

Our model shares many objectives of other group memory projects, including the need for a maintenance activity separate from day-to-day use [Berlin et al. 93; Terveen, Selfridge, Long 93]. However, our emphasis on domain orientation sets our approach apart. In particular, domain orientation is an interesting perspective from which to view two major challenges for shared and evolving information spaces: (1) the development of classification conventions that support information location, and (2) the ability to actively deliver information to users when it is relevant to their task at hand [Fischer et al. 93].

Systems designed for general information storage and retrieval face the difficult task of developing information categories that make sense to the individuals who share the information [Berlin et al. 93]. General categorization schemes are dependent on the group members that develop and use them, and therefore will change as group members come and go. Design domains, on the other hand, are characterized by domain-specific conventions that have relatively precise and stable meaning to domain practitioners. Domain conventions have developed over time to enable designers to conceptualize design problems and to communicate important ideas. The relative stability of domain conventions make domain-oriented systems less sensitive to turnover in group personnel.

General-purpose information spaces can have only a limited notion of the user’s task at hand. Domain-oriented design environments exploit domain semantics and the design context [Fischer et al. 93] to actively notify designers when there is information about which they should know. Active information delivery helps designers to detect inconsistencies early in the design process, and to learn about design concepts of which they were unaware.

9. Summary

This chapter has described a process model for the evolution of domain-oriented design environments through use. We consider design environments as seeds that grow by accumulating design knowledge as they are used to support design tasks. Periodically, a reseeding process is necessary to ensure that new knowledge is accessible to the design environment’s computational mechanisms and therefore is accessible to designers using the environment. We claim that such an approach is necessary to support design in complex and open-ended domains, in which new design knowledge surfaces in the context of design tasks.

A seed is a collection of knowledge and procedures capable of growing — of sustaining growth — through interaction with domain designers during day-to-day use. It stimulates, focuses, and mediates discussion — and thus knowledge capture — during the incremental growth phase. The seed must be capable of capturing the information elicited from the use of the system. There is no absolute requirement for the completeness, correctness, or specificity of the information in the seed. In fact, it is often its shortcomings in these respects that provoke input from designers.

Evolutionary growth during system use is a process of adding information related directly or indirectly to the artifact being designed. Thus, the artifact (in our case, the network logical map) is the foundation for evolutionary growth. During the growth phase the designers who use the system are primarily focused on their task at hand. Information input is highly situation specific — tied to a specific artifact and stated in particular rather than in general. For a while, information grows in an orderly manner, but eventually order breaks down and the system begins to degrade in usefulness.

Reseeding is necessary when evolutionary growth stops proceeding smoothly. During reseeding, the system's information is restructured, generalized, and formalized to serve future design tasks. The reseeding process creates a forum to discuss what design information captured in the context of specific design projects should be incorporated into the extended seed to support the next cycle of evolutionary growth and reseeding. Tools contained in design environments support reseeding by making suggestions about how the information can be formalized.

10. Acknowledgments

The authors would like to thank the members of the Human-Computer Communication group at the University of Colorado, who contributed substantially to the conceptual framework and the systems discussed in this paper. In particular, Kumiyo Nakakoji provided invaluable assistance. The research was supported by the National Science Foundation under grants No. IRI-9015441 and MDR-9253245, and NYNEX Science and Technology Center (White Plains, N.Y.).

11. References

[Anderson 85]
J.R. Anderson, *Cognitive Psychology and Its Implications (2nd Edition)*, W.H. Freeman and Co., New York, 1985.

[Berlin et al. 93]
L. Berlin, R. Jeffries, V.L. O'Day, A. Paepcke, C. Wharton, *Where Did You Put It? Issues in the Design and Use of a Group Memory*, Human Factors in Computing Systems, INTERCHI'93 Conference Pr oceedings, ACM, 1993, pp. 23-30.

[Buchanan, Shortliffe 84]
B.G. Buchanan, E.H. Shortliffe, *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley Publishing Company, Reading, MA, 1984.

[Carraher, Carraher, Schliemann 85]
T.N. Carraher, D.W. Carraher, A.D. Schliemann, *Mathematics in the streets and in the schools*, British Journal of Developmental Psychology, Vol. 3, 1985, pp. 21-29.

[ChenDietterichUllman 91]
A. Chen, T.G. Dietterick, D.G. Ullman, *A Computer-Based Design History Tool*, Proceedings of the 1991 NSF Design and Manufactering Systems Conference, SME (Society of Manufactering Engineers), Austin, Texas, January 1991, pp. 985-994.

[Conklin, Begeman 88]
J. Conklin, M. Begeman, *gIBIS: A Hypertext Tool for Exploratory Policy Discussion*, Transactions of Office Information Systems, Vol. 6, No. 4, October 1988, pp. 303-331.

[CSTB 90]
Computer Science and Technology Board, *Scaling Up: A Research Agenda for Software Engineering*, Communications of the ACM, Vol. 33, No. 3, March 1990, pp. 281-293.

[DeMarco, Lister 87]
T. DeMarco, T. Lister, *Peopleware: Productive Projects and Teams*, Dorset, New York, 1987.

[Ellis, Gibbs, Rein 91]
C.A. Ellis, S.J. Gibbs, G.L. Rein, *Groupware: Some Issues and Experiences*, Communications of the ACM, Vol. 34, No. 1, 1991, pp. 38-58.

[Fischer 92]
G. Fischer, *Domain-Oriented Design Environments*, Proceedings of the 7th Annual Knowledge-Based Software Engineering (KBSE-92) Conference (McLean, VA), IEEE Computer Society Press, Los Alamitos, CA, September 1992, pp. 204-213.

[Fischer et al. 91]
G. Fischer, A.C. Lemke, R. McCall, A. Morch, *Making Argumentation Serve Design*, Human Computer Interaction, Vol. 6, No. 3-4, 1991, pp. 393-419.

[Fischer et al. 92]
G. Fischer, J. Grudin, A.C. Lemke, R. McCall, J. Ostwald, B.N. Reeves, F. Shipman, *Supporting Indirect, Collaborative Design with Integrated Knowledge-Based Design Environments*, Human Computer Interaction, Special Issue on Computer Supported Cooperative Work, Vol. 7, No. 3, 1992, pp. 281-314.

[Fischer et al. 93]
G. Fischer, K. Nakakoji, J. Ostwald, G. Stahl, T. Sumner, *Embedding Computer-Based Critics in the Contexts of Design*, Human Factors in Computing Systems, INTERCHI'93 Conference Proceedings, ACM, 1993, pp. 157-164.

[Fischer, Girgensohn 90]
G. Fischer, A. Girgensohn, *End-User Modifiability in Design Environments*, Human Factors in Computing Systems, CHI'90 Conference Proceedings (Seattle, WA), ACM, New York, April 1990, pp. 183-191.

[Fischer, Lemke 88]
G. Fischer, A.C. Lemke, *Construction Kits and Design Environments: Steps Toward Human Problem-Domain Communication*, Human-Computer Interaction, Vol. 3, No. 3, 1988, pp. 179-222.

[Fischer, McCall, Morch 89a]
G. Fischer, R. McCall, A. Morch, *Design Environments for Constructive and Argumentative Design*, Human Factors in Computing Systems, CHI'89 Conference Proceedings (Austin, TX), ACM, New York, May 1989, pp. 269-275.

[Fischer, McCall, Morch 89b]
G. Fischer, R. McCall, A. Morch, *JANUS: Integrating Hypertext with a Knowledge-Based Design Environment*, Proceedings of Hypertext'89 (Pittsburgh, PA), ACM, New York, November 1989, pp. 105-117.

[Fischer, Nakakoji 91]
G. Fischer, K. Nakakoji, *Making Design Objects Relevant to the Task at Hand*, Proceedings of AAAI-91, Ninth National Conference on Artificial Intelligence, AAAI Press/The MIT Press, Cambridge, MA, 1991, pp. 67-73.

[Girgensohn 92]
A. Girgensohn, *End-User Modifiability in Knowledge-Based Design Environments*, Ph.D. Dissertation, Department of Computer Science, University of Colorado, Boulder, CO, 1992, Also available as TechReport CU-CS-595-92.

[Grudin 94a]
J. Grudin, *Evaluating Opportunities for Design Capture*, in T. Moran and J. Carroll (eds.), *Design Rationale: Concepts, Techniques, and Use*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1994, (in press).

[Grudin 94b]
J. Grudin, *Computer-Supported Cooperative Work: History and focus*, IEEE Computer, Vol. 27, No. 5, 1994, pp. 19-26.

[Hackman, Kaplan 74]
J.R. Hackman, R.E. Kaplan, *Interventions into group process: An approach to improving the effectiveness of groups*, Decision Sciences, Vol. 5, 1974, pp. 459-480.

[Hill 89]
W.C. Hill, *The Mind at AI: Horseless Carriage to Clock*, AI Magazine, Vol. 10, No. 2, Summer 1989, pp. 29-41.

[Hill et al. 92]
W.C. Hill, J.D. Hollan, D. Wroblewski, T. McCandless, *Edit Wear and Read Wear*, Human Factors in Computing Systems, CHI'92 Conference Proceedings (Monterrey, CA), ACM, May 1992, pp. 3-9.

[Hollan, Stornetta 92]
J. Hollan, S. Stornetta, *Beyond Being There*, Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems, ACM, New York, 1992, pp. 119-125.

[Hutchins 90]
E. Hutchins, *The Technology of Team Navigation*, in P. Galegher, R. Kraut, C. Egido (ed.), *Intellectual Teamwork*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1990, ch. 8.

[Johansen 88]
R. Johansen, *Groupware: Computer Support for Business Teams*, The Free Press, New York, 1988.

[Kuffner, Ullman 91]
T.A. Kuffner, D.G. Ullman, *The information requests of mechanical design engineers*, Design Studies, Vol. 12, No. 1, January 1991, pp. 42-50.

[Kunz, Rittel 70]
W. Kunz, H.W.J. Rittel, *Issues as Elements of Information Systems*, Working Paper 131, Center for Planning and Development Research, University of California, Berkeley, CA, 1970.

[Lave 88]
J. Lave, *Cognition in Practice*, Cambridge University Press, Cambridge, UK, 1988.

[Lee 92]
A. Lee, *Investigations into History Tools for user Support*, Unpublished Ph.D. Dissertation, University of Toronto, 1992.

[Lemke, Fischer 90]
A.C. Lemke, G. Fischer, *A Cooperative Problem Solving System for User Interface Design*, Proceedings of AAAI-90, Eighth National Conference on Artificial Intelligence, AAAI Press/The MIT Press, Cambridge, MA, August 1990, pp. 479-484.

[McCall 91]
R. McCall, *PHI: A Conceptual Foundation for Design Hypermedia*, Design Studies, Vol. 12, No. 1, 1991, pp. 30-41.

[McCall et al. 90]
R. McCall, P. Bennett, P. d'Oronzio, J. Ostwald, F. Shipman, N. Wallace, *PHIDIAS: Integrating CAD Graphics into Dynamic Hypertext*, European Conference on Hypertext (ECHT '90), 1990, pp. 152-165.

[McCall, Ostwald, Shipman 91]
R. McCall, J. Ostwald, F. Shipman, *Supporting Designers' Access to Information Through Virtually Structured Hypermedia*, Proceedings of the 1991 Conference on Intelligent Computer Aided Design, Elsevier, New York, NY, 1991, pp. 116-127.

[McCall, Schaab, Schuler 83]
R. McCall, B. Schaab, W. Schuler, *An Information Station for the Problem Solver: System Concepts*, in C. Keren, L. Perlmutter (eds.), *Applications of Mini- and Microcomputers in Information, Documentation and Libraries*, Elsevier, New York, 1983.

[Nakakoji 93]
K. Nakakoji, *Increasing Shared Understanding of a Design Task Between Designers and Design Environments: The Role of a Specification Component*, Unpublished Ph.D. Dissertation, Department of Computer Science, University of Colorado, 1993, Also available as TechReport CU-CS-651-93.

[Nardi 93]
B.A. Nardi, *A Small Matter of Programming*, The MIT Press, Cambridge, MA, 1993.

[Polanyi 66]
M. Polanyi, *The Tacit Dimension*, Doubleday, Garden City, NY, 1966.

[Reder 82]
L.M. Reder, *Plausability judgment versus fact retrieval: Alternative strategies for sentence verification*, Psychological Review, Vol. 89, 1982, pp. 250-280.

[Reeves 93]
B.N. Reeves, *The Role of Embedded Communication and Artifact History in Collaborative Design*, Ph.D. Dissertation CU-CS-694-93, Department of Computer Science, University of Colorado, Boulder, CO, 1993.

[Reeves, Shipman 92a]
B.N. Reeves, F. Shipman, *Supporting Communication between Designers with Artifact-Centered Evolving Information Spaces*, Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'92), ACM, New York, November 1992, pp. 394-401.

[Reeves, Shipman 92b]
B.N. Reeves, F. Shipman, *Making it Easy for Designers to Provide Design Rationale,* Working Notes of the AAAI 1992 Workshop on Design Rationale Capture and Use, AAAI, San Jose, CA, July 1992, pp. 227-233.

[Repenning, Sumner 92]
A. Repenning, T. Sumner, *Using Agentsheets to Create a Voice Dialog Design Environment*, Proceedings of the 1992 ACM/SIGAPP Symposium on Applied Computing, ACM Press, 1992, pp. 1199-1207.

[Schoen 83]
D.A. Schoen, *The Reflective Practitioner: How Professionals Think in Action*, Basic Books, New York, 1983.

[Shipman 93]
F. Shipman, *Supporting Knowledge-Base Evolution with Incremental Formalization*, Ph.D. Dissertation, Department of Computer Science, University of Colorado, Boulder, CO, 1993, Also available as TechReport CU-CS-658-93.

[Shipman, McCall 94]
F. Shipman, R. McCall, *Supporting Knowledge-Base Evolution with Incremental Formalization*, Human Factors in Computing Systems, INTERCHI'94 Conference Proceedings, ACM, 1994, pp. 285-291.

[Stefik et al. 87]
M. Stefik, G. Foster, D.G. Bobrow, K. Kahn, S. Lanning, L. Suchman, *Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings*, Communications of the ACM, Vol. 30, No. 1, January 1987, pp. 32-47.

[Suchman 87]
L.A. Suchman, *Plans and Situated Actions*, Cambridge University Press, Cambridge, UK, 1987.

[Terveen, Selfridge, Long 93]
L.G. Terveen, P.G. Selfridge, M.D. Long, *From Folklore to Living Design Memory*, Human Factors in Computing Systems, INTERCHI'93 Conference Proceedings, ACM, April 1993, pp. 15-22.

[Thimbleby, Anderson, Witten 90]
H. Thimbleby, S. Anderson, I.H. Witten, *Reflexive CSCW: Supporting Long-Term Personal Work*, Interacting with Computers, Vol. 2, No. 3, 1990, pp. 330-336.

[Walker et al 87]
J.H. Walker, David A. Moon, Daniel L. Weinreb, Mike McMahon, *The Symbolics Genera Programming Environment*, IEEE Software, Vol. 4, No. 6, November 1987, pp. 36-45.

[Yakemovic, Conklin 90]
K.C. Yakemovic, E.J. Conklin, *Report of a Development Project Use of an Issue-Based Information System*, Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'90), 1990, pp. 105-118.

Table of Contents

List of Figures